AD-A202038

RSRE MEMORANDUM No. 4193

ROYAL SIGNALS & RADAR ESTABLISHMENT

A HYBRID OPTIMISATION STRATEGY FOR ADAPTIVE FEED-FORWARD LAYERED NETWORKS

Authors: A Webb & D Lowe

[illegible]

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
R S R E MALVERN,
WORCS.

88 12 22 038

RSRE MEMORANDUM No. 4193

- A -

Royal Signals and Radar Establishment
Memorandum 4193

A Hybrid Optimisation Strategy for Adaptive Feed-forward Layered Networks

Andrew Webb & David Lowe
Speech Research Unit, SP4

9th September 1988.

British document

Abstract

This ~~work~~ considers an optimisation strategy for solving for the weight values of a general adaptive feed-forward layered network with linear output units. The weight values of the final layer of the network are determined using linear techniques based on a singular value decomposition of the outputs of the hidden units. The set of weight values governing the transformation of the data prior to this final layer is found using a nonlinear optimisation strategy. This memorandum considers various nonlinear optimisation strategies combined with the linear method and compares the performance of this hybrid approach with previous work ~~[7]~~ which solves for all the weights of the network using nonlinear techniques on a range of problems.

Contents

1 Introduction.

Adaptive feed–forward networks are a particular example of a wider class of neural networks, which are networks of computing structures operating in parallel whose configurations are motivated by biological systems. Specifically, a network is an inhomogeneous assembly of simple computational elements connected by links with variable weights. The networks perform a pattern classification through a nonlinear, noninvertible mapping from pattern space to class space. The precise form of this transformation is determined by the configuration of the network and the values of the weights between individual elements in the network. The configuration of the network is often influenced by *a priori* knowledge of the pattern classification problem that the network is required to perform. Once this *structure* has been specified, the values of the parameters of the network, the weights and biases, are found through a training procedure conditional upon a set of training patterns. Ideally, the network will capture the structure of the training data, thus giving it some ability to generalise to previously unseen patterns.

The training procedure usually requires the minimisation of an expression for the error at the output of the network. Traditionally, this minimisation has been implemented using 'error back propagation' (the chain rule) to evaluate the derivatives of the error with respect to the network parameters, together with a modified steepest descent method to use the gradient information to find a local minimum of the error function [5]. Other more efficient strategies for minimising a nonlinear function have been considered for adaptive feed–forward networks. These methods have been applied to examples ranging from small-scale trivial problems (solution of the XOR function) to a larger scale pattern processing problem (speech recognition of isolated confusable words) [7].

The aim of this paper is to present a combined linear and nonlinear technique for solving for the set of weights of the network. This technique takes explicit note of the fact that, providing the output transfer functions are linear (or more generally, that the output nonlinearity is invertible) then, given the weights between the input layer and the hidden layer, the weights between the hidden layer and the output layer may be chosen, using linear techniques, to find a global minimum of the error function with respect to these weights. The first layer weights may be found using the techniques described in our previous report [7].

The condition that the final layer units have linear transfer functions is not so restrictive as may first appear. It has been shown [6] that minimising the mean–square error at the output of such a network is equivalent to performing a nonlinear discriminant analysis which maximises a norm relating the inverse of the total covariance matrix and the weighted between class covariance matrix of the hidden unit patterns. Therefore, in problems where the maximisation of such a norm is applicable, the addition of a final layer with linear output units and sum–square error minimisation enables this to be achieved.

The outline of this paper is as follows. We begin by describing the standard feed–forward adaptive network and the procedure for training and testing such a network. The following section considers the special case where the output transfer function of the network (see Section 2) is linear, which enables linear techniques to be used to solve for the final-layer weights. We combine the linear techniques of Section 3 and nonlinear optimisation methods in Section 4 to present a strategy for solving for the weights of a network. This hybrid linear-nonlinear strategy is applied to a range of problems in Section 5 and the results compared

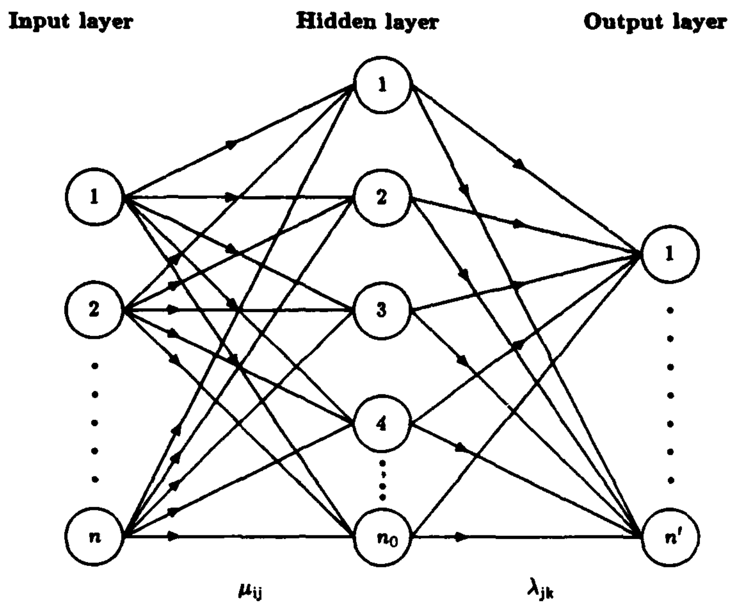

Figure 1: A schematic diagram of the standard feed forward adaptive layered network geometry considered in this paper.

with the standard nonlinear optimisation approach.

2 The Standard Feed–forward Adaptive Network.

2.1 Description of the Network.

The structure of the standard layered network model considered in this paper is depicted in Figure 1. It is envisaged that input data may be represented by an arbitrary (real valued) n-dimensional vector, $|x\rangle$[1], or an ordered sequence of n real valued numbers, $\{x_i;\ i = 1, \ldots, n\}$. Thus there are n independent input nodes to the network which accept each input data vector. Each input node is totally connected to a set of n_0 'hidden' nodes (hidden from direct interaction with the environment). Associated with each link between the i-th input node and the j-th hidden node is a scalar μ_{ij}. Usually, the fan-in to a hidden node takes the form of a hyperplane: the input to node j is of the form $\theta_j = \sum_{i=1}^{n} x_i\mu_{ij} = \langle x|\mu_j\rangle$ where

[1]This paper employs the bra–ket notation for vectors. A column vector $(x_1, x_2, \ldots)$ is written as $|x\rangle$ (the 'ket'). The corresponding row vector is denoted $\langle x|$ (the 'bra' vector). A scalar product between $|x\rangle$ and $\langle y|$ is given by $\langle y|x\rangle$ and $|y\rangle\langle x|$ is a linear operator with matrix elements $A_{ij} = y_i x_j$.

$|\mu_j\rangle$ is the vector of n scalar values associated with hidden node j. The rôle of each hidden node is to accept the value provided by the fan-in and output a value obtained by passing it through a (generally, though not necessarily) nonlinear transfer function,

$$\phi_j = \phi(\mu_{0j} + \theta_j) = \phi_j(\mu_{0j} + \langle x|\mu_j\rangle) \tag{1}$$

where μ_{0j} is a local 'bias' associated with each hidden node.

The hidden layer is fully connected to a set of n' output nodes corresponding to the components of an n' dimensional output space. The strength of the connection from the j-th hidden node to the k-th output node is denoted λ_{jk} and thus the value received at the k-th output node is a weighted sum of the output values from all of the hidden nodes, $I_k = \sum_{j=1}^{n_0} \lambda_{jk}\phi_j$.

In general the output from the k-th output node will be a nonlinear function of its input, $O_k = \Phi_k(\lambda_{0k} + \langle\lambda_k|\phi\rangle)$ where λ_{0k} is a 'bias' associated with that output node.

For a Radial Basis Function network [1] the fan-in to a hidden node takes the form of a hypersphere; $\theta_j = \sum_{i=1}^{n}(x_i - \mu_{ij})^2$ without a bias link ($\mu_{0j} = 0 \ \forall\ j$) and the output from the output nodes are linear functions of the input ($\Phi_k(x) = x \ \forall\ k$). Apart from these superficial differences, we view the essential rôle of these networks to be equivalent in spite of the interpretational distinctions.

Thus the networks provide a transformation mapping from an n-dimensional input space to an n'-dimensional output space via an intermediate characterisation space. This mapping is totally defined by the topology of the network (in particular, how many hidden units are employed) once all the nonlinear transfer functions are specified and the set of weights and biases $\{\lambda, \mu\}$ have been determined. This set of weights and biases is found by a 'training' procedure.

Networks performing a transformation from an n-dimensional input space to an n'-dimensional output space using more than one intermediate hidden layer have been considered by some workers [4], but we shall restrict our attention in this paper to networks with a single hidden layer.

2.2 Training the Network.

The network will operate once a set of weight values $\{\lambda_{jk},\ \mu_{ij}\}$ has been determined. This set is conditional upon training data presented in the form of representative input and corresponding target output patterns.

The set of parameters $\{\lambda_{jk},\ \mu_{ij}\}$ is chosen so that the actual outputs of the network, $\{|O^p\rangle, p = 1, 2, \ldots P\}$, for a given set of inputs, $\{|I^p\rangle, p = 1, 2, \ldots P\}$, are 'close' in some sense to the desired target values, $\{|T^p\rangle, p = 1, 2, \ldots P\}$. Usually, this error criterion is a sum-of-squares error of the form

$$E = \sum_{p=1}^{P} \| \ |T^p\rangle - |O^p\rangle \ \|^2 \tag{2}$$

where the summation runs over all the patterns in the training set. This is the error measure we shall adopt in this paper. Using the Euclidean distance function and expressing the outputs in terms of the set of weights and biases and the inputs, the error may be written explicitly as a function of the set $\{\lambda_{jk},\ \mu_{ij}\}$. For instance, in the case of the standard multi-layer perceptron, this error may be expressed as

$$E = \sum_{p=1}^{P} \sum_{k=1}^{n'} \left\{ T_k^p - \Phi_k \Big(\lambda_{0k} + \sum_{j=1}^{n_0} \phi_j [\mu_{0j} + \sum_{i=1}^{n} I_i^p \mu_{ij}] \lambda_{jk} \Big) \right\}^2 \tag{3}$$

If the transformation mapping was a Radial Basis Function network, the explicit error expression would be

$$E = \sum_{p=1}^{P} \sum_{k=1}^{n'} \left\{ T_k^p - \Big(\lambda_{0k} + \sum_{j=1}^{n_0} \phi_j [\sum_{i=1}^{n} (I_i^p - \mu_{ij})^2] \lambda_{jk} \Big) \right\}^2 \tag{4}$$

However, in what follows we will restrict our attention to the multi-layer perceptron.

The expression for the error is a differentiable nonlinear function of the parameters and the training procedure is to find a minimum of this function. Therefore, some strategy for nonlinear function minimisation must be employed. Of course, a global minimum cannot be guaranteed. Nevertheless, it may be possible to obtain a good local minimum. Also, not only do we require a good solution, but it must be obtained 'within a reasonable timescale'. Schemes which find a good solution a small percentage of the time, but are very fast, may be preferable to one which finds a good solution on most occasions, but takes a long time to do so.

Optimisation strategies for nonlinear functions were discussed in a previous paper [7]. These were applied to the training of adaptive feed-forward networks and various example problems considered. The optimisation strategies considered in some detail were two examples of a conjugate gradient method (the Fletcher-Reeves (F-R) and the Polak-Ribiere (P-R) variants); two examples of the Broyden class of quasi-Newton methods (the Davidon-Fletcher-Powell (DFP) and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods); and a nonlinear sum-of-squares minimisation technique (the Levenberg-Marquardt (LM) method). Further variants of the methods arose from the type of line search employed to find the minimum of the nonlinear function in a given search direction. Two types of line search were used; an approximate line search based on inverse parabolic interpolation and an accurate line search which performed inverse parabolic interpolation iteratively and resulted in less iterations per solution, but more function evaluations. The performance of these methods was compared with that of the steepest descent method traditionally used for training an adaptive feed-forward network.

The results of the study showed that the most suitable optimisation strategy depended on the size of the network under consideration. For small-scale problems (typically less than 30 adjustable parameters), the Levenberg-Marquardt method gave the best results. For medium-scale problems (between 40 and 250 adjustable parameters), the quasi-Newton methods (BFGS and DFP) offered the greatest efficiency. For large-scale problems (with the number of parameters, N, much greater than 250) the conjugate gradient method was adopted since we were unable to apply the Levenberg-Marquardt or quasi-Newton methods owing to the requirement to store an array of size $N \times N$. Steepest descents was not recommended for any real problem due to its inefficiency.

A further result was that there is little advantage to be gained by using an accurate line minimisation procedure. An accurate line search increased the CPU time by about a factor of three for the problems addressed without a significant improvement in classification performance. As a result of the study, the optimisation strategies we shall consider in Section 5 are conjugate gradients (P–R and F–R), the quasi-Newton methods (BFGS and DFP) and the Levenberg–Marquardt method, all using an approximate line search.

2.3 Testing the Network.

Testing the network consists of applying the trained network to patterns not previously used as part of the training set and comparing the outputs with the labels corresponding to those patterns. It is not sufficient to consider how closely the network models the training set since, if it models the training set too well, the network may not have captured the underlying structure of the data and be unable to generalise to unseen data.

The data employed for testing the network consists of a set of P' input-output presentations $\{|\tilde{I}^p\rangle, |\tilde{T}^p\rangle, p = 1, 2, \ldots P'\}$. For the set of input test patterns, $\{|\tilde{I}^p\rangle\}$, the outputs produced by the trained network, $\{|\tilde{O}^p\rangle\}$, are compared with the desired outputs, $\{|\tilde{T}^p\rangle\}$ using a sum-squared-error measure. We use the normalised error presented in our previous paper [7], $\mathcal{E}$

$$\mathcal{E} = \sqrt{\frac{\sum_{i=1}^{P'} \| |\tilde{T}^i\rangle - |\tilde{O}^i\rangle \|^2}{\sum_{i=1}^{P'} \| |\tilde{T}^i\rangle - |\bar{T}\rangle \|^2}} \tag{5}$$

where $|\bar{T}\rangle$ is the mean target pattern over all the test patterns. For a normalised error of unity, one can infer that the network is predicting the output *in the mean* and a value of zero means that it is predicting perfectly. In the results of Section 5, the ability of the chosen networks to generalise to unseen data is indicated by the normalised error, averaged over each set of experiments.

3 Linear Output Units.

In this section, we consider the special case where the output nonlinearity is described by $\Phi(x) = x$, that is, the transfer function on each output unit is linear. Of course, there may be situations where it is desirable for the output transfer function to have the logistic form $\Phi(x) = 1/(1 + exp(-x))$, particularly where knowledge of the output space suggests that values between 0 and 1 are appropriate. On the other hand, having linear output units enables the weights between the hidden layer and the output layer, $\{\lambda\}$, to be calculated explicitly in terms of the weights between the input layer and hidden layer, $\{\mu\}$, as follows.

The error between the kth output of the network and the kth target value (for the pth pattern presented at the input), E_k^p, may be written as

$$E_k^p = O_k^p - T_k^p. \tag{6}$$

Expressing the output, O_k^p, as a function of the weights, $\{\mu, \lambda\}$, the nonlinearities, ϕ_j, and the inputs, I_j^p, gives

$$E_k^p = \lambda_{0k} + \sum_{j=1}^{n_0} \phi_j[\mu_{0j} + \sum_{i=1}^{n} I_i^p \mu_{ij}]\lambda_{jk} - T_k^p. \tag{7}$$

In matrix notation, Equation (7) becomes

$$\mathbf{E} = \mathbf{\Lambda H} - \mathbf{T} \tag{8}$$

where $\mathbf{E}$ and $\mathbf{T}$ are $n' \times P$ matrices, $(\mathbf{E})_{jp} = E_j^p$ and $(\mathbf{T})_{jp} = T_j^p$; $\mathbf{\Lambda}$ is a $n' \times (n_0+1)$ matrix of weights and biases,

$$\begin{aligned}(\mathbf{\Lambda})_{ji} &= \lambda_{ij} & i &= 1,\ldots,n_0,\\ &= \lambda_{0j} & i &= n_0+1,\end{aligned} \tag{9}$$

and $\mathbf{H}$ is a $(n_0+1) \times P$ matrix of the outputs of the hidden units

$$\begin{aligned}(\mathbf{H})_{jp} &= \phi_j[\mu_{0j} + \textstyle\sum_{i=1}^{n} I_i^p \mu_{ij}] & j &= 1,\ldots,n_0,\\ &= 1 & j &= n_0+1.\end{aligned} \tag{10}$$

The solution of Equation (8) for $\mathbf{\Lambda}$ which minimises $||\mathbf{E}||$, the Euclidean norm of the matrix $\mathbf{E}$, and which has the smallest $||\mathbf{\Lambda}||$ may be written as

$$\mathbf{\Lambda} = \mathbf{TH}^+, \tag{11}$$

where $\mathbf{H}^+$ is the pseudo-inverse of $\mathbf{H}$ of size $P \times (n_0+1)$ (see Appendix A). For a matrix $\mathbf{H}$ of rank n_0+1, the pseudo-inverse is given by

$$\mathbf{H}^+ = \mathbf{H}^*(\mathbf{HH}^*)^{-1} \tag{12}$$

where $\mathbf{H}^*$ denotes the transpose of matrix $\mathbf{H}$. The condition on the rank of the matrix $\mathbf{H}$ implies that the number of independent patterns, P, is greater than n_0+1. In a linear problem, this corresponds to an overdetermined problem which does not have a unique solution generally. Substituting for $\mathbf{H}^+$ from Equation (12) into Equation (11) gives the solution for the weight matrix, $\mathbf{\Lambda}$, as

$$\mathbf{\Lambda} = \mathbf{TH}^*(\mathbf{HH}^*)^{-1} \tag{13}$$

Thus, given the set of weights between the input layer and the hidden layer, $\{\mu\}$, Equation (11) gives the solution for the final layer weights which minimises the output error. There are two immediate consequences of this approach. Firstly, for a given set $\{\mu\}$, the solution for $\mathbf{\Lambda}$ gives a global minimum of the error as a function of the weights $\{\lambda\}$. This means that it is not necessary to search the space of final layer weights since Equation (11) provides a scheme for obtaining these weights from the first layer weights. Secondly, expressing the final layer weights $\{\lambda\}$ as a function of the first layer weights $\{\mu\}$ reduces the number of independent variables of the error function. For some network configurations, this may be a significant reduction, though it must be recognised that any savings due to fewer independent variables may be offset by the need to calculate a pseudo-inverse.

It is worth considering the significance of this approach in a little more detail. The error at the output of the network, E, is equivalent to the trace of the matrix, $\mathbf{EE}^*$. Using Equation (8) for $\mathbf{E}$ and the minimum norm solution for $\mathbf{\Lambda}$ (Equation (11)) gives

$$E = Tr(\mathbf{EE}^*) = Tr(\mathbf{TT}^* - \mathbf{TH}^*(\mathbf{HH}^*)^+\mathbf{HT}^*) \tag{14}$$

Thus, for a fixed set of target vectors, minimising the sum-square error, E, is identical to maximising the cost function, C, given by

$$C = Tr(\mathbf{TH}^*\mathbf{S}^+\mathbf{HT}^*), \tag{15}$$

where $\mathbf{S} = \mathbf{H}\mathbf{H}^*$ is the $(n_0 + 1) \times (n_0 + 1)$ covariance matrix at the output of the hidden units.

The interpretation of the cost function, C, for different output coding schemes in terms of the total covariance matrix and the between class covariance matrix of hidden unit outputs is given in reference [6]. However, we shall note here that for a one–out–of–n' output coding, and if the matrix $\mathbf{S}$ is full rank, then minimising the sum–square error at the output of the network is equivalent to maximising a cost function, C, given by

$$C = Tr(\mathbf{S}_B \mathbf{S}^{-1}), \tag{16}$$

the trace of the product of the weighted between–class covariance matrix $\mathbf{S}_B$ and the inverse of the total covariance matrix at the outputs of the final hidden layer. If there are n' classes, $C_k, k = 1, ..., n'$ with n_k patterns in class C_k, then for this particular coding scheme the matrix $\mathbf{S}_B$ is given by

$$\mathbf{S}_B = \sum_{k=1}^{n'} n_k^2 (|m_k^H\rangle - |m^H\rangle)(\langle m_k^H| - \langle m^H|) \tag{17}$$

where $|m_k^H\rangle$ is the mean output vector over all patterns in class C_k (at the hidden unit outputs)

$$|m_k^H\rangle = \frac{1}{n_k} \sum_{|\phi^p\rangle \in C_k} |\phi^p\rangle \tag{18}$$

and $|m^H\rangle$ is the mean output vector over all patterns at the hidden unit outputs

$$|m^H\rangle = \frac{1}{P} \sum_{p=1}^{P} |\phi^p\rangle. \tag{19}$$

Equation (16) justifies the use of linear output units and the minimisation of the sum–square–error since the process implicitly maximises the cost function (16) which is attempting to separate the pattern into classes in an optimal way. This is the correct strategy for discrimination tasks.

In the following section we shall present a strategy for finding the weights of an adaptive feed-forward network using a combination of linear and nonlinear optimisation techniques.

4 Combined Linear and Nonlinear Optimisation.

In Section 2 we described a standard feed-forward adaptive network and presented a scheme for 'training' the network. Traditionally, all the parameters of the network have been determined by minimising the output error as a function of the parameters using a steepest descent, or accelerated steepest descent strategy. In our previous work [7], more efficient nonlinear function optimisation routines were assessed and applied to error minimisation at the output of a feed-forward network on a range of problems. The previous section described how, for the special case of linear output units, the final layer weights may be chosen optimally, using linear techniques, as a function of the first layer weights. We shall now consider combining the linear and nonlinear optimisation strategies.

4.1 Training the Network.

Again, the set of parameters, $\{\lambda, \mu\}$, is chosen to minimise the sum–of–squares error, E, which may be written (for linear output units) as

$$E = \sum_{p=1}^{P} \sum_{k=1}^{n'} \left\{ T_k^p - \lambda_{0k} + \sum_{j=1}^{n_0} \phi_j [\mu_{0j} + \sum_{i=1}^{n} I_i^p \mu_{ij}] \lambda_{jk} \right\}^2 . \qquad (20)$$

We now regard E as a nonlinear function of the first layer weights $\{\mu\}$ only and a nonlinear function optimisation strategy is employed to find these weights. The second layer weights, $\{\lambda\}$, are regarded as a function of the first layer weights and are chosen using Equation (11). Thus, each time the nonlinear strategy updates the first layer weights (for example, when performing a line search along a chosen search direction) the second layer weights are also updated. This search strategy may be considered as proceeding on two timescales. A long timescale over which the set of weights $\{\mu\}$ is adjusted in order to minimise the error, and a very short timescale in which the set of weights $\{\lambda\}$ is changed to minimise the error as a function of these weights alone.

Exploiting the linearity of the transfer function at the outputs in the way described offers a number of potential advantages in the training of a network. The number of independent weights in the network is reduced. Therefore, it might be expected that the time taken for the nonlinear function optimisation scheme to find a minimum of the error will be reduced. The network is always in a state where the error is at a global minimum in the space of final layer weights. This may help the network to reach a minimum more rapidly and to reach a shallow local minimum less often. However, any reduction in the number of iterations to find a minimum may be offset by the extra computation involved in evaluating the singular value decomposition, particularly if the number of final layer weights is large.

In the following section we consider applying a selection of nonlinear optimisation strategies to several problems ranging from the small-scale XOR problem to a large-scale speech recognition experiment.

4.2 Testing the Network.

The testing of the network is independent of the method used to obtain the weight values. Patterns not previously used as part of the training set are presented to the network and the outputs compared with the targets corresponding to those patterns (see Section 2.3). A measure of the performance of the network is the normalised error (Equation (5)). Evaluated on the training data, it indicates how well that data is modelled by the network. Evaluated on the test data, it shows how well the network has captured the underlying structure of data and is a measure of the ability of the network to generalise.

For classification problems (for example, the 'EE' set example considered in the next section), a further measure of performance is the classification accuracy. For a one–from–n' output coding scheme, patterns are assigned to class C_j if the actual output vector is closest to the desired target pattern for that class. It does not necessarily follow that the smaller the normalised error, the better the classification accuracy, as the results of Section 5 show.

The linear analysis suggests a further measure of classification accuracy. Since minimising the mean–square–error implies that a particular cost function, evaluated on the hidden unit outputs, is maximised, then classification may be performed at the output of the hidden units by assuming a model for the distribution of patterns. This may be achieved by assuming a multivariate Gaussian distribution for the outputs of the hidden units for each class, with the means and covariance matrices of the distributions taken to be the sample class means and covariance matrices at the outputs of the hidden units. These may be calculated using the set of input patterns and the solution for the network weights. A previously unseen pattern may be classified by presenting it to the network, calculating the output of the hidden units and assigning it to a particular class using the probability density functions on a maximum likelihood basis. This strategy will not be followed in this study, but its importance is noted.

In the following section, the normalised error will be used as a measure of performance and, where appropriate, the number of patterns classified correctly based on the smallest Euclidean distance in the output space.

5 Applications and Results.

In Section 3 a linear method for evaluating the weights of the final layer of an adaptive feed-forward layered network (under certain reasonable assumptions) was described. In Section 4, it was shown how a network may be trained using a combination of linear and nonlinear optimisation strategies. Here, this hybrid approach is applied to the five problems addressed in our previous work [6].

The network considered is a conventional multi-layer perceptron with one hidden layer whose units have logistic transfer functions and a linear output layer. The nonlinear optimisation strategies employed are two variants of the conjugate gradients method (Polak-Ribiere and Fletcher-Reeves); two quasi-Newton methods (the Broyden-Fletcher-Goldfarb-Shanno and Davidon-Fletcher-Powell methods) and the Levenberg-Marquardt method. The method of steepest descents was not used since in our experience it generally takes an order of magnitude longer to converge to a solution than the optimisation strategies listed above. An approximate line search was used in all the methods and to test for convergence, a fractional tolerance, *ftol*, was employed. Iteration will stop if

$$2.0 \times |S - S_p| \leq ftol \times (|S| + |S_p| + \epsilon) \qquad (21)$$

where S is the current function value; S_p is the value on the previous iteration and ϵ is a small positive constant (taken to be 10^{-10}). The value of the fractional tolerance used in the convergence criterion was taken to be 10^{-5} unless otherwise stated. The singular value decomposition routine used to implement the linear part of the hybrid optimisation strategy is based on a Householder reduction to bi-diagonal form followed by diagonalisation of the bi-diagonal form using Givens rotations [3].

The tables of results are in the same format as those in our previous work [7]. For each problem a solution is sought for a number of different initial random configurations for the weights of the network. These are chosen from a uniforn distribution on $(-1, 1)$. The tables give the number of correct solutions (where 'correct' will be defined for each problem separately); the mean number of iterations of the algorithm (which is equal to the number of gradient evaluations of the mean-square error); the mean normalised error (evaluated on the test set if one is used, otherwise the training set); the mean number of function evaluations (calculations of the mean-square error); and the mean CPU time in seconds (all experiments were run on a microvax II running FORTRAN under VAX/VMS). The mean CPU time can be taken only as a rough guide to the performance, since the algorithms have not been optimised. In the tables 'LIN' refers to the hybrid approach employing a linear strategy to solve for the weights between the hidden layer and final layer. 'STD' refers to the standard approach in which all weights are solved for using a nonlinear optimisation routine (but still with a final layer with linear transfer functions).

5.1 The XOR function.

The XOR function is a small-scale problem consisting of only four input patterns, namely the binary sequences 00, 01, 10, 11, and the target for the patterns is a '1' if the number of '1's in the input is odd and '0' otherwise. The XOR experiment consisted of presenting the four input patterns, together with the corresponding desired target patterns to a network with two input units, two logistically nonlinear hidden units and one linear output unit. For an initial random weight configuration, the chosen search strategy is employed to find the minimum of the output error. This is repeated for 1000 different sets of random start positions. Results are summarised in Table 1, and (as in [7]) a 'correct' solution was assumed if the normalised error on the training data was less than 0.05.

In the fully nonlinear approach, there are 9 parameters to be adjusted by the nonlinear optimisation routine. In the hybrid approach, there are 6, since the two weights and the bias between the hidden layer and the final layer are found using linear techniques.

From Table 1 we may conclude that

1. Using linear techniques to solve for the final layer weights results in an increase in performance for the quasi-Newton methods and a marginal decrease in the performance of conjugate gradient methods. The performance of the Levenberg-Marquardt method is substantially reduced from that obtained by the fully nonlinear method with only 76% correct compared with 94% for the standard approach. However, Levenberg-Marquardt still gives the best performance overall.
2. For all methods the number of function calls and gradient calls is significantly reduced. For the Levenberg-Marquardt method, this reduction is by a factor of two but is between four and seven for the other methods.
3. The mean normalised error is reduced for all methods except the Levenberg-Marquardt method.
4. Despite the great reduction in the number of function and gradient calls achieved by the hybrid approach, the mean CPU time is essentially unaltered. This is because of the additional requirement to calculate a pseudo-inverse each time the function is evaluated.

5.2 Markov Source Discriminator.

The Markov source discriminator is an intermediate problem between the small-scale discontinuous problems and large, continuous, stochastic problems. It is a generalisation of

	% Correct	Number of iterations	Mean Error, $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
LIN & C.G. (P-R)	66.5	5.7	0.2418	37	0.48
LIN & C.G. (F-R)	65.7	5.7	0.2459	39	0.50
LIN & q-N (BFGS)	70.1	6.2	0.2187	45	0.59
LIN & q-N (DFP)	70.6	6.4	0.2080	46	0.61
LIN & L-M	76.0	5.8	0.1722	31	0.52
STD & C.G. (P-R)	67.4	40	0.2802	239	0.75
STD & C.G. (F-R)	67.5	43	0.2835	241	0.78
STD & q-N (BFGS)	62.8	26	0.2928	155	0.59
STD & q-N (DFP)	47.3	56	0.4157	391	1.36
STD & L-M	94.0	10	0.0432	54	0.46

C.G = conjugate gradients, q-N = quasi-Newton, L-M = Levenberg-Marquardt
P-R and P-R refer to the Polak-Ribiere and Fletcher-Reeves implementations of the conjugate gradient method.
DPF and BFGS refer to the Davidon-Fletcher-Powell and Broyden-Fletcher-Golfarb-Shanno quasi-Newton methods
LIN and STD refer to the method for solving for the weights (using linear techniques or the standard approach)

Table 1: Results Table illustrating the performance of various search strategies on the XOR function problem using a 2-2-1 standard layered network with linear output units.

the XOR function described above. Each input pattern is a short binary first-order Markov chain; i.e. a sequence of zeros and ones in which the probability of a one in any position depends on the symbol at the previous position. Such a pattern is generated by a symmetric Markov source. A symmetric Markov source is completely specified by a transition probability, p, of any bit being different from the previous bit.

Results are presented for six-bit patterns generated by two equally-likely Markov sources with transition probabilities of $p = 0.7$ and $p = 0.3$ (the classic exclusive-OR problem considered in the previous section is the limiting case of two-bit patterns and transition probabilities of 0.0 and 1.0).

Training consisted of presenting the 64 (2^6) distinct six-bit input patterns to the network together with the associated targets for a given set of initial random start weights. The target for each pattern was the probability that the pattern was generated by a binary first-order Markov source with transition $p = 0.7$ (see [7]) and the network considered was $6 - n_0 - 1$ where we chose $n_0 = 6$. The procedure was repeated for one hundred sets of start weights and performed for each search strategy. The criterion that the solution obtained was 'correct' was taken to be when the normalised error at the convergence of the algorithm was below 0.25. For this experiment, the fractional tolerance used to test for convergence was set to 10^{-4}.

The number of independent parameters in the fully nonlinear approach is 49. For the hybrid approach, this is reduced to 42.

From Table 2 we may conclude that

1. There is no improvement to be gained in using the hybrid scheme for this problem. All methods give fewer correct solutions.
2. The number of iterations taken to find a correct solution is reduced for the quasi-Newton methods, but is increased for the Levenberg-Marquardt and conjugate gradient methods. The number of function calls is reduced for all methods except the Levenberg-Marquardt method, for which it is increased by almost a factor of two.
3. The time taken to find a correct solution for the hybrid approach is between two and three times that taken for the full nonlinear optimisation.

5.3 Chaotic Time Series.

The time series considered is the *quadratic map* where the value of the time series at time $t+1$, x^{t+1}, is given in terms of the value at time t by

$$x^{t+1} = 4x^t(1 - x^t) \tag{22}$$

This map is known to be chaotic on the interval $[0, 1]$. The autocorrelation function of this time series may be shown to be delta distribution correlated [1]. Thus, although the time series may appear to be intrinsically extremely complicated, it is completely deterministic and has been generated by a smooth, continuous and differentiable map (a quadratic functional).

	% Correct	Number of iterations	Mean error $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
LIN & C.G. (P-R)	88	306	0.1506	1294	651
LIN & C.G. (F-R)	69	328	0.2281	1304	664
LIN & q-N (BFGS)	97	183	0.1265	885	435
LIN & q-N (DFP)	94	173	0.1397	979	479
LIN & L-M	99	228	0.1031	1290	1166
STD & C.G. (F-R)	92	283	0.2161	1654	217
STD & C.G. (P-R)	96	272	0.1643	1659	217
STD & q-N (BFGS)	100	258	0.0919	1163	188
STD & q-N (DFP)	100	187	0.1004	1005	153
STD & L-M	100	140	0.0667	671	524

C.G = conjugate gradients,
q-N = quasi-Newton, L-M = Levenberg-Marquardt,

Table 2: Results Table illustrating the performance of various search strategies on the Markov Source Discriminator using a 6–6–1 standard layered network with linear output units.

The experiment performed is one in which the network attempts to predict the mapping given by Equation (22). The input training pattern is a data point in the interval $(0, 1)$. The desired target pattern is the number obtained from the mapping given by (22). Training involved presenting 100 input–output pairs of real numbers, where the input numbers were distributed according to a uniform random measure on the interval $(0, 1)$. For a given initial random configuration of the weights, each search strategy was used to minimise the output error. Once the network had been trained, it was tested on a different set of 100 random patterns chosen over the same interval and the normalised error calculated. This was repeated for 100 different random start configurations of the network weights. For this problem, a solution was determined as 'correct' if the normalised error on the test set was less than 0.01. For the fully nonlinear optimisation scheme, the number of parameters is 19, whilst for the hybrid approach the number of parameters in the nonlinear optimisation algorithm is 12.

From Table 3 we may infer

1. For this problem, the hybrid approach gives superior performance compared to the general nonlinear strategy. All hybrid methods obtain 100% correct solutions.
2. The number of iterations and function calls is reduced markedly (by a factor of between 10 and 40).
3. For the hybrid approaches, the mean CPU time *per iteration* is about four seconds, compared with only one second for the fully nonlinear minimisation. This is due to the requirement to calculate the pseudo–inverse. However, because of the significant reduction in the number of iterations, the mean CPU time *per solution* is also reduced, though not so dramatically.

	% Correct	Number of iterations	Mean Error, $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
LIN & C.G. (P-R)	100	4.7	0.0019	29	15
LIN & C.G. (F-R)	100	4.7	0.0021	29	15
LIN & q-N (BFGS)	100	4.8	0.0020	31	16
LIN & q-N (DFP)	100	4.4	0.0021	30	15
LIN & L-M	100	5.6	0.0012	38	23
STD & C.G. (P-R)	59	192	0.1679	1285	202.5
STD & C.G. (F-R)	70	135	0.1013	838	133.7
STD & q-N (BFGS)	97	143	0.0303	788	131.3
STD & q-N (DFP)	95	110	0.0044	720	115.4
STD & L-M	100	58	0.0016	341	109.2

C.G = conjugate gradients,
q-N = quasi-Newton, L-M = Levenberg-Marquardt,

Table 3: *Results Table illustrating the performance of various search strategies on the* quadratic map problem for a 1–6–1 standard layered network with linear output units.

5.4 Point-source Location.

The specific problem considered in this subsection is the estimation of the position of a single source in the far field of an imaging system, given its sampled image vector. A vector is generated from the outputs of a 4 × 4 array of elements in the focal-plane of an imaging system (giving an image vector of dimension 16) when the scene contains a single point source. By moving the source around the scene and measuring its position and the outputs of the array, a library of image vectors is generated, together with corresponding source positions. Training a network consists of presenting images of a point source at different positions in the far field as input with the corresponding azimuth-elevation positions of the source as targets. The image vectors are normalised and the input data comprises the normalised images of the source at positions which lie on a 21 × 21 grid covering the field of view of the array (see [7] for a more complete description of the imaging problem). The target data used for training are the positions of the sources which give rise to the image. Thus, for the training phase, there are 441 patterns of dimension 16 for the inputs and dimension 2 for the targets. For the test data, the normalised images of a single source at 400 positions which are randomly distributed across the field of view are taken as input with the positions as targets. Figure 2 shows the calibration source positions for the training data, the source positions for the test data and the sensor positions.

The network chosen to represent the mapping has 12 logistically nonlinear hidden units and 2 linear output units, giving 230 independent weights for the fully nonlinear optimisation strategy and 204 for the hybrid strategy. For the point-source location problem, the criterion chosen as to whether a 'correct' solution was obtained was that the normalised error on test should be less than 0.015. The experiment was run for 10 different random start configurations for the network weights. The results are shown in Table 4.

From Table 4 we may conclude

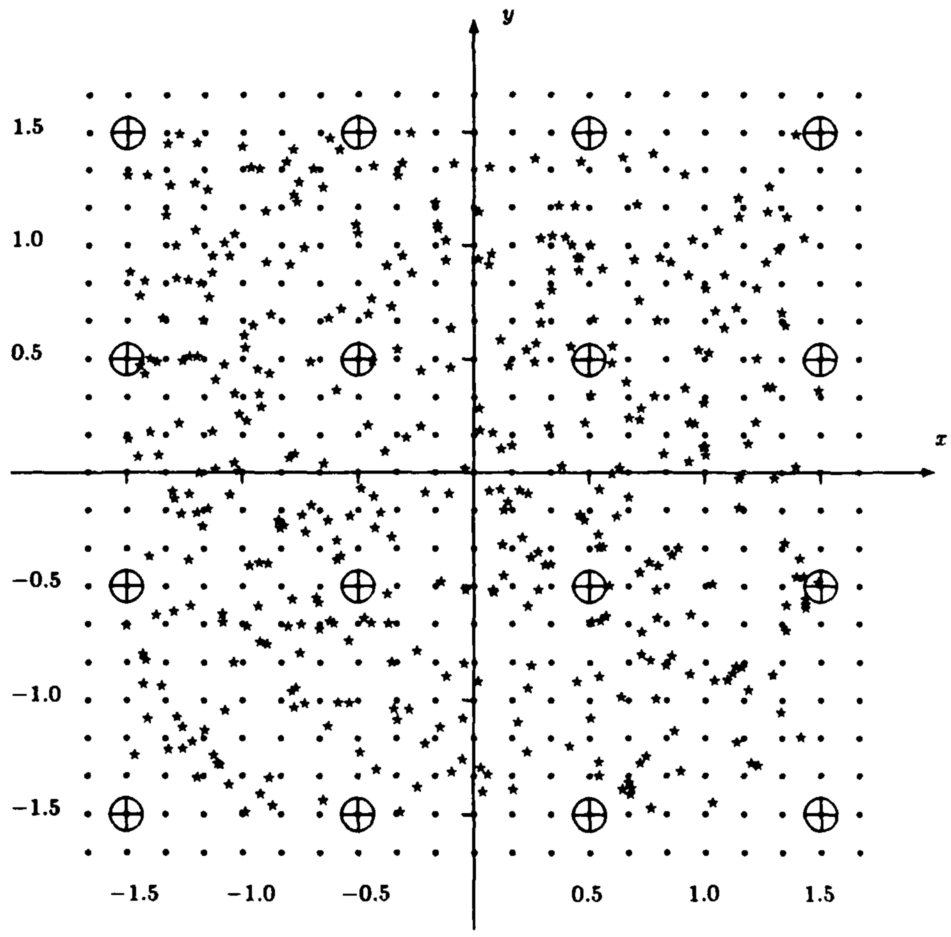

Figure 2: Diagram showing the calibration target positions for the training data (circles), the target positions for the test data (stars) and the sensor positions (circled crosses) plotted in dimensionless units.

	% Correct	Number of iterations	Mean Error, $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
LIN & C.G. (P-R)	100	526	0.0052	2019	19712
LIN & C.G. (F-R)	100	967	0.0039	3858	38150
LIN & q-N (BFGS)	100	465	0.0023	2105	19708
LIN & q-N (DFP)	100	219	0.0043	1118	10671
LIN & L-M	100	174	0.0017	944	54341
STD & C.G. (P-R)	100	1582	0.0058	9030	23260
STD & C.G. (F-R)	100	1930	0.0051	11076	28414
STD & q-N (BFGS)	100	1908	0.0010	7775	26800
STD & q-N (DFP)	100	613	0.0022	3260	9549
STD & L-M	100	110	0.0016	604	37493

S.D = steepest descent, C.G = conjugate gradients,
q-N = quasi-Newton, L-M = Levenberg-Marquardt,

Table 4: Results Table showing the performance of the search strategies on the synthetic Focal Plane Array problem using a 16-12-2 standard layered network with linear output units (230 weights to adjust).

1. Both the fully nonlinear optimisation and the hybrid approach achieve 100% correct solutions.
2. For the conjugate gradients and the quasi-Newton methods, the hybrid approach reaches a minimum of the error in a fewer number of iterations than the standard methods. Typically, this reduction is about a factor of three. Surprisingly, the number of iterations increases for the Levenberg-Marquardt method.
3. The mean errors are reduced for the conjugate gradient methods and are increased for the quasi-Newton methods. The error for the Levenberg-Marquardt is effectively unaltered.
4. The mean CPU time *per iteration* is greater for the hybrid approach. However, the mean CPU time *per solution* depends on the particular variant of the conjugate gradient or quasi-Newton method. The mean CPU time per solution is greater for the hybrid approach for one of the conjugate gradient implementations (F-R) and one of the quasi-Newton methods (DFP) and is reduced for the others.
5. For both the hybrid and standard approaches, the mean CPU time for the Levenberg-Marquardt method is greatest.

5.5 Speech Recognition: the 'EE' set.

This problem is a speaker-dependent speech recognition experiment where the words to be classified are the highly confusable eight 'EE' sounds from the alphabet, i.e. {'B', 'C', 'D', 'E', 'G', 'P', 'T', 'V'}. A single speaker produced 40 utterances of each of the eight target sounds which were converted into time-frequency patterns with 40 time slices and 19

run number	Number of iterations	Training Error, $\mathcal{E}$	Testing Error, $\mathcal{E}$	Training /160	Testing /160	Number of function calls	CPU (seconds)
1 LIN	45	0.37800	0.5072	140	132	202	3026
2 LIN	71	0.37797	0.4885	140	137	288	4421
3 LIN	91	3.2×10^{-6}	0.3382	160	152	355	5435
4 LIN	363	0.0017	0.5121	160	132	1669	24460
5 LIN	123	1.0×10^{-6}	0.3049	160	152	475	7414
1 STD	411	0.3791	0.5205	140	128	1946	25530
2 STD	465	0.3793	0.4756	140	137	1924	26220
3 STD	2726	0.0036	0.4226	160	144	12340	164000
4 STD	2180	0.0029	0.3764	160	150	10140	134000
5 STD	484	0.3791	0.5039	144	143	2013	27510

C.G = conjugate gradients, q–N = quasi–Newton, L–M = Levenberg–Marquardt

Table 5: Results Table illustrating the performance of various search strategies on EE set problem for a 760–8–8 standard layered network with linear output units.

frequency channels as described in [7]. Twenty sets of the eight patterns were used to train the network and the remaining sets of patterns were used to test the ability of the network to generalise. The desired target output patterns were taken to be a one–from–eight coding. Thus the number of input units to the network was 760, the number of hidden units was taken to be 8 and the number of output units was also 8. In this experiment the total number of parameters for the fully nonlinear approach is 6160. For the hybrid approach there are 6088 parameters in the nonlinear part.

In this experiment, a solution was determined as correct if the actual output vector of the network was closest to the desired target vector. Table 5 gives results for a conjugate gradient search strategy for five different random starts for the initial weights of the network.

The table shows

1. In all five experiments the number of iterations is greatly reduced by using the hybrid approach. There is a corresponding reduction in the number of function evaluations.

2. The CPU time for the hybrid approaches is also much lower than the standard, fully nonlinear strategy.

3. On four of the five examples there is a greater number of correct solutions on test for the hybrid approach.

4. Two of the five experiments for the hybrid approach produce normalised errors on train of less than 10^{-5}, showing that the training data was modelled almost exactly. These two experiments also gave the smallest two normalised errors on the test set and the greatest number of correct solutions on the test set.

6 Discussion.

This memorandum has considered a hybrid strategy for solving for the set of unknown parameters in an adaptive feed–forward layered network such as a multi–layer perceptron. The strategy has been to solve for the parameters in each layer of the network separately, which ought to be a method of preventing sensitive errors produced in one layer propagating to subsequent layers (as the chain rule would indicate). In addition, since the transfer functions on the output nodes were chosen to be linear, the adjustable parameters between the final layer of hidden units and the output nodes (i.e. the links and biases) may be determined by a linear optimisation technique.

This strategy may be viewed in two ways : (1) parameters in different layers may be considered to vary on different time scales, the hidden–output weights are 'slaved' to the behaviour of the input–hidden units; (2) the parameters of the hidden–output layer are implicit functions of the input–hidden weights. The latter interpretation implies that the number of adjustable parameters in a network may be reduced, and thus a nonlinear search strategy may be performed in a reduced–dimension subspace which ought to be more efficient. At least for linear output transfer functions, solving for the final layer weights in terms of other parameters *guarantees* a globally optimum solution in the space spanned by the final layer weights.

This strategy has been applied to the class of problems considered in our previous memorandum [7]. The nonlinear optimisation of the weights between the input and hidden layers was performed by one of the several efficient search methods described therein and the linear optimisation of the final layer weights was performed using a singular value decomposition to produce a pseudo–inverse of the matrix of outputs of the hidden units.

On the basis of our study, we can infer that in almost all instances the number of iterations required to reach a solution was significantly reduced. However, the mean CPU time *per iteration* increased. This is not significant since it is a consequence of evaluating the pseudo–inverse at every iteration. The pseudo–inverse is not essential : any (iterative) technique could be employed to solve for the (linear) least–mean–square error at the final layer which may be faster, though the minimum norm solution to the minimum error would not be guaranteed.

What is significant, however, is that for some combinations of techniques, the percentage of correct solutions was degraded in the hybrid technique compared to solving for all the network weights simultaneously by the same nonlinear optimisation method. This is surprising and unexpected. A tentative suggestion for this trend is that for some types of problem combined with efficient search techniques, restricting the search procedures to operate in a lower dimensional weight space allows a higher percentage of 'poor' local minima to be found. This may be circumvented if the search method was allowed to move in high dimensions and to flow around the 'barrier'. However, in other solutions (for example, the chaotic time series) the hybrid technique is far superior, in terms of the number of iteration required and the percentage of 'good' solutions found. It is interesting to note that in the largest scale network addressed (the 'EE' set) only the hybrid approach succeeded in finding a zero–error solution on training on the few random weight starts considered. This is possible in this particular example because the experiment is underdetermined – there are not sufficient patterns to estimate all the adjustable parameters reliably and so the network is capable of fitting the training data exactly. Normally this would imply that generalisation performance should be poor. However, these zero-error networks also produced the smallest generalisation errors and maximum classification performance (95% correct on the unseen data). This observation, combined with the result that the mean normalised error across all experiments is generally less in the hybrid approach than the fully nonlinear approach, suggests that solving for the network weights layer by layer tends to produce lower minima, thus leading to a more efficient optimisation strategy.

The results of this memorandum show that a hybrid approach generally leads to a minimisation of the error at the output of a network in fewer iteration compared with a fully nonlinear optimisation strategy. For certain problems (e.g. high-dimensional, confusable, inherently noisy data illustrated by the 'EE' set) the results suggest that it is more efficient to train a network (according to least-squares minimisation) by using a hybrid technique which solves for the network parameters layer by layer.

Appendix A Singular Value Decomposition

Any $M \times N$ matrix, $\mathbf{A}$ (it is assumed that $M \geq N$), can be written in the form [3]

$$\mathbf{A} = \mathbf{U\Sigma V^*} \tag{23}$$

where $\mathbf{U^*U} = \mathbf{V^*V} = \mathbf{VV^*} = \mathbf{I}_N$, the $N \times N$ identity matrix, and $*$ denotes the transpose of a matrix. The matrix $\mathbf{\Sigma} = diag(\sigma_1, \sigma_2, ..., \sigma_N)$ is a diagonal matrix whose elements are the non-negative square-roots of the eigenvalues of $\mathbf{A^*A}$. These are the ***singular values*** of $\mathbf{A}$ and may be ordered so that

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N \geq 0 \tag{24}$$

The $M \times N$ matrix $\mathbf{U}$ consists of the N orthonormalised eigenvectors associated with the N largest eigenvalues of $\mathbf{AA^*}$, and the $N \times N$ matrix $\mathbf{V}$ consists of the orthonormalised eigenvectors of $\mathbf{A^*A}$. The decomposition given by Equation (23) is termed the ***singular value decomposition*** of the matrix $\mathbf{A}$.

If $\mathbf{A}$ is of rank $R \leq N$, then $\sigma_{R+1} = \sigma_{R+2} = \dots = \sigma_N = 0$ and Equation (23) may be written

$$\mathbf{A} = \mathbf{U}_R\mathbf{\Sigma}_R\mathbf{V}_R^* \tag{25}$$

with $\mathbf{U}_R^*\mathbf{U}_R = \mathbf{V}_R^*\mathbf{V}_R = \mathbf{I}_R$ and $\mathbf{\Sigma}_R = diag(\sigma_1, \dots, \sigma_R)$.

The ***pseudo-inverse*** of a $M \times N$ matrix $\mathbf{A}$, is the $N \times M$ matrix, $\mathbf{A}^+$, which satisfies the four properties: $\mathbf{AA^+A} = \mathbf{A}$, $\mathbf{A^+AA^+} = \mathbf{A^+}$, $(\mathbf{AA^+})^* = \mathbf{AA^+}$ and $(\mathbf{A^+A})^* = \mathbf{A^+A}$. The pseudo-inverse can always be determined and is unique. It can be computed from the singular value decomposition of $\mathbf{A}$ by

$$\mathbf{A}^+ = \mathbf{V}_R\mathbf{\Sigma}_R^+\mathbf{U}_R^* \tag{26}$$

where $\mathbf{\Sigma}_R^+ = diag(1/\sigma_1, 1/\sigma_2, \dots, 1/\sigma_R)$. The pseudo-inverse provides a solution to the least-squares problem,

$$\mathbf{A}|x\rangle = |b\rangle \tag{27}$$

Of all the vectors $|x\rangle$ which minimise the sum-of-squares $(\langle b| - \langle x|\mathbf{A}^*)(|b\rangle - \mathbf{A}|x\rangle)$, the minimum norm solution (i.e. the one with the smallest $\|x\|^2 = \langle x|x\rangle$) is given by

$$|x\rangle = \mathbf{A}^+|b\rangle \tag{28}$$

References

[1] Broomhead, D.S. and Lowe, David,: "Radial Basis Functions, Multi-variable Functional Interpolation and Adaptive Networks", RSRE Memo. 4148, March 1988.

[2] Devijver, P.A. and Kittler, J.: "Pattern Recognition A Statistical Approach", Prentice-Hall International, Inc., London, 1982.

[3] Golub, G.H. and Reinsch, C.: "Singular Value Decomposition and Least Squares Solutions", in "Handbook for Automatic Computation. Vol. 2: Linear Algebra", Wilkinson, J.H. and Reinsch, C. (eds), Springer-Verlag, 1971.

[4] Peeling, S.M. and Moore, R.K.: "Experiments in Isolated Digit Recognition Using the Multi-layer Perceptron", RSRE Memo. 4073, December 1987.

[5] Rumelhart, D.E., Hinton, G.E. and Williams, R.J.: "Learning Internal Representation by Error Propagation", in "Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations.", Rumelhart, D.E. and McClelland (eds), Bradford Books/MIT Press,

[6] Webb, A.R. and Lowe, David.: "A Theorem Connecting Adaptive Feed-forward Layered Networks and Nonlinear Discriminant Analysis", RSRE Memo. 4209, August 1988.

[7] Webb, A.R., Lowe, David, and Bedworth, M.D.: "A Comparison of Nonlinear Optimisation Strategies for Feed-forward Adaptive Layered Networks", RSRE Memo. 4157, July 1988.

DOCUMENT CONTROL SHEET

Overall security classification of sheet ...UNCLASSIFIED...

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference Memorandum 4193	3. Agency Reference	4. Report Security Classification Unclassified

5. Originator's Code (if known) 7784000	6. Originator (Corporate Author) Name and Location Royal Signals and Radar Establishment St Andrews Road, Malvern, Worcestershire WR14 3PS
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title
A HYBRID OPTIMISATION STRATEGY FOR ADAPTIVE FEED-FORWARD LAYERED NETWORKS

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
Webb A R	Lowe D		9.88	22

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement
Unlimited

Descriptors (or keywords)

continue on separate piece of paper

Abstract

This work considers an optimisation strategy for solving for the weight values of a general adaptive feed-forward layered network with linear output units. The weight values of the final layer of the network are determined using linear techniques based on a singular value decomposition of the outputs of the hidden units. The set of weight values governing the transformation of the data prior to this final layer is found using a nonlinear optimisation strategy. This Memorandum considers various nonlinear optimisation strategies combined with the linear method and compares the performance of this hybrid approach with previous work which solves for all the weights of the network using nonlinear techniques on a range of problems.

S80/48